Boolean Operations on Surfaces for VTK

Release 0.00

Bryn Lloyd[1]

April 29, 2010

[1]Computer Vision Laboratory, ETH Zurich

Abstract

This document describes an implementation of a VTK wrapper for the GNU Triangulated Surface library GTS. This paper is accompanied with the source code, and the xml files necessary to create a paraview plugin.

Contents

A recurring question on the vtk-users list is the ability to perform Boolean operations (union, intersection and difference) with surface meshes. Currently, this is not possible using VTK, except possibly for very simple specific cases.

1 Methods

Boolean operations are implemented in the GNU Triangulated Surface library http://gts.sourceforge.net/. It uses exact geometric predicates adapted from code by Jonathan Shewchuk in order to find exact intersections between triangles of different surfaces.

We provide a new class implemented in the VTK style called vtkSurfaceBooleanOperations. It requires two input connections (on the same port), each passing a surface. Internally we convert the surfaces (vtkPolyData) into a GTS surface object and back using the functions:

```
void vtk2gts (vtkPolyData * input, GtsSurface * output);
void gts2vtk (GtsSurface * input, vtkPolyData * output);
```

The actual intersection tests are performed using the GTS surfaces:

```
  GNode *tree1 = gts_bb_tree_surface(surface1);
  GNode *tree2 = gts_bb_tree_surface(surface2);

  GtsSurfaceInter* inter = gts_surface_inter_new(gts_surface_inter_class(),
    surface1,
    surface2,
    tree1,
    tree2,
    !gts_surface_is_closed(surface1),
    !gts_surface_is_closed(surface2)
  );

  GtsSurface *surface = gts_surface_new(gts_surface_class(),
                                gts_face_class(),
                                gts_edge_class(),
                                gts_vertex_class());

  if(this->Union) {
    gts_surface_inter_boolean(inter, surface, GTS_1_OUT_2);
    gts_surface_inter_boolean(inter, surface, GTS_2_OUT_1);
  } else if(this->Intersection) {
    gts_surface_inter_boolean (inter, surface, GTS_1_IN_2);
    gts_surface_inter_boolean (inter, surface, GTS_2_IN_1);
  } else if(this->Difference) {
    gts_surface_inter_boolean (inter, surface, GTS_1_OUT_2);
    gts_surface_inter_boolean (inter, surface, GTS_2_IN_1);
    gts_surface_foreach_face (inter->s2,
            (GtsFunc) gts_triangle_revert, NULL);
    gts_surface_foreach_face (surface2, (GtsFunc) gts_triangle_revert, NULL);
  }
```

It can be seen that the intersection is done using oriented bounding box trees. It can also be seen that the difference operator is not symmetric, i.e. it depends on which surface is surface 1 and which is surface 2.

2 Results

An example application is provided (example-usage.cxx).

```
#include "vtkSurfaceBooleanOperations.h"

#include <vtkPolyData.h>
#include <vtkSphereSource.h>
#include <vtkPolyDataWriter.h>

#include <vtkSmartPointer.h>
#define vtkNew(type,name) \
  vtkSmartPointer<type> name = vtkSmartPointer<type>::New()
```

```
int main(int argc, char ** argv) {

  vtkNew(vtkSphereSource,source1);
  source1->SetCenter(0.0, 0.0, 0.0);
  source1->SetRadius(1.0);

  vtkNew(vtkSphereSource,source2);
  source2->SetCenter(0.5, 0.0, 0.0);
  source2->SetRadius(1.0);


  vtkNew(vtkSurfaceBooleanOperations,booleanOperator);
  booleanOperator->AddInputConnection(source1->GetOutputPort());
  booleanOperator->AddInputConnection(source2->GetOutputPort());

  vtkNew(vtkPolyDataWriter,writer);
  writer->SetInputConnection(booleanOperator->GetOutputPort());

  booleanOperator->SetUnionOn();
  writer->SetFileName("Union.vtk");
  writer->Update();

  booleanOperator->SetIntersectionOn();
  writer->SetFileName("Intersection.vtk");
  writer->Update();

  booleanOperator->SetDifferenceOn();
  writer->SetFileName("Difference.vtk");
  writer->Update();

  return 0;
}
```

It performs all three boolean operations and saves the result as VTK legacy files. The input to the filter is two overlapping sphere surfaces generated via vtkSphereSource. The results are shown in the figure below.

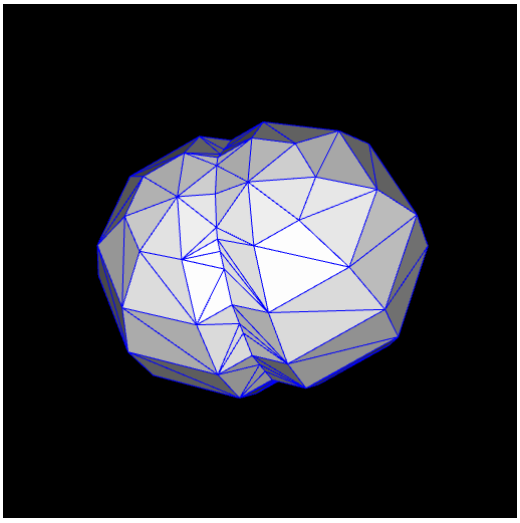

Figure 1. This figure shows the union of two spheres generated via intersection of two surface meshes.

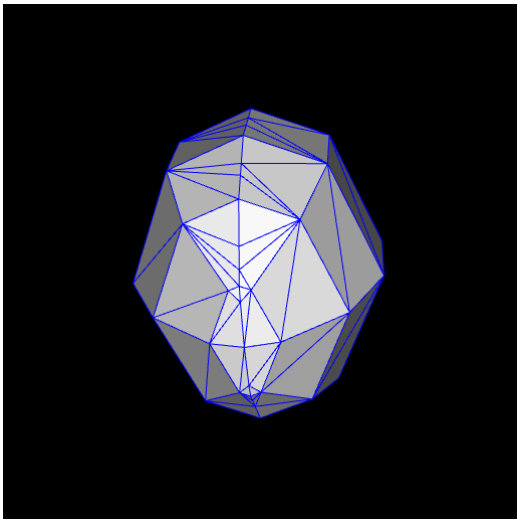

Figure 2. This figure shows the intersection of two spheres with radius 1, and centers at (0,0,0) and (1,0,0).

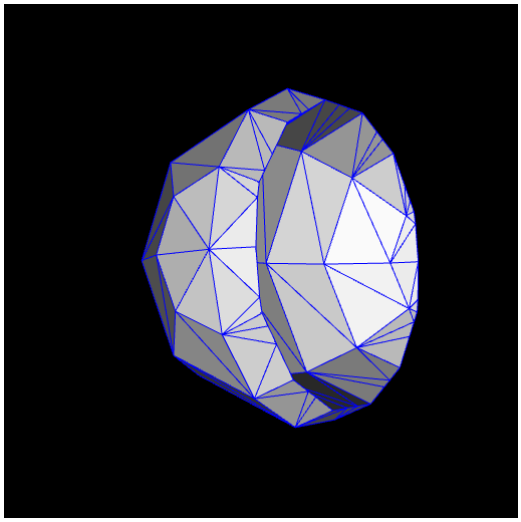

Figure 3. This figure shows the difference between surface 1 and surface 2.

3 Software Requirements

You need to have the following software installed:

- VTK 5.2 or newer.
- CMake 2.6 or newer
- GTS and GLIB

We provide a copy of the GTS and GLIB libraries, which are compiled as part of the configuration using cmake. This may take a while. We have tested the configuration and compilation process on a Linux machine and expect it to work also on Mac OS X (gts is available through the fink project http://www.finkproject.org/). GTS and GLIB are available under the GPL license.

The CMakeLists.txt will perform the compilation each time you configure using cmake. If the compilation was successful the first time you configure, then set the cmake variable `COMPILE_GTS` to `OFF`.